N O V A S - C

Naval Observatory Vector Astrometry Subroutines
C Language Version 2.0

J. A. Bangert
U. S. Naval Observatory

Based on algorithms and Fortran code by:
G. H. Kaplan
U. S. Naval Observatory

Version 1.0: 7 June 1996
Version 2.0: 1 December 1998

CONTENTS

NOVAS-C
Naval Observatory Vector Astrometry Subroutines
C Language Version 2.0

John A. Bangert
Astronomical Applications Department
U. S. Naval Observatory

Based on the algorithms and Fortran code by:
George H. Kaplan
Astronomical Applications Department
U. S. Naval Observatory

1.0 Introduction

The **N**aval **O**bservatory **V**ector **A**strometry **S**ubroutines, NOVAS, is an integrated package of source-code modules for computing a wide variety of common astrometric quantities and transformations. The package can provide, in one function call, the instantaneous coordinates (apparent, topocentric, or astrometric place) of any star or planet. At a lower level, NOVAS also provides general astrometric utility transformations, such as those for precession, nutation, aberration, parallax, and the gravitational deflection of light. The computations are highly precise. The NOVAS algorithms are, in fact, virtually identical to those now used in the production of *The Astronomical Almanac*. NOVAS is easy-to-use and can be incorporated into data reduction programs, telescope control systems, and simulations.

The first version of NOVAS was released in 1988 as a package of Fortran subroutines [1]. The Fortran package proved to be very popular, but there were requests for a C-language version of the software. In the early 1990s, members of the U.S. Naval Observatory (USNO)/Naval Research Laboratory Optical Interferometer group converted parts of NOVAS to C for use in their project. Their work was returned to USNO's Astronomical Applications Department for further development. This work lead to the first complete edition of NOVAS in ANSI-standard C (designated NOVAS-C Version 1.0), released in 1996. A major revision of the NOVAS Fortran code took place in 1998, with the primary goal of supporting data conforming to the International Celestial Reference System (ICRS) [2]. Shortly thereafter, NOVAS-C was updated to reflect the changes in the Fortran code and to add additional capabilities. The result was NOVAS-C Version 2.0, described in this document.

NOVAS-C uses, as input, astrometric reference data that is expressed in the International Astronomical Union (IAU) J2000.0 system. In particular, NOVAS-C 2.0 supports (but is not limited to) data that conforms to the ICRS. ICRS-compatible data includes the Hipparcos and Tycho Catalogues [3], the ACT Reference Catalog [4], the International Celestial Reference Frame (ICRF) [5], the Jet Propulsion Laboratory's

DE405 planetary ephemeris [6], and Earth orientation measurements from the International Earth Rotation Service (IERS) [7]. The list of ICRS-compatible data of various types is continually expanding. NOVAS-C can also be used with data conforming to the FK5 system.

In addition to support for data conforming to the ICRS, NOVAS-C 2.0 also provides direct support for USNO/AE98—USNO's new fundamental ephemerides of selected minor planets[8]. Furthermore, NOVAS-C can be easily modified to use other ephemerides as well.

The algorithms used by the NOVAS-C functions are based on a vector and matrix formulation that is rigorous, consistent with recent IAU resolutions, and does not use spherical trigonometry or form "day numbers" at any point. Objects within and outside the solar system are treated similarly and the position vectors formed and operated on by these functions place each relevant object at its actual distance (in AU) from the solar system barycenter. Objects at unknown distance (parallax zero or undetermined) are placed on the "celestial sphere" herein defined to be at a radius of 10 megaparsecs (2.06×10^{12} AU). A description of the algorithms used in NOVAS-C, along with definitions of terms and related information, can be found in [9]. A few very minor revisions to the algorithms were made in 1998 for compliance with the ICRS system. See also [10] for an evaluation of the precision of the NOVAS algorithms that involve relativity.

NOVAS-C contains three levels of functions: basic, utility, and supervisory. *Basic*-level functions supply the values of fundamental variables, such as the nutation angles and the heliocentric positions of solar system bodies, for specific epochs. *Utility*-level functions perform transformations corresponding to precession, nutation, aberration, etc. *Supervisory*-level functions call the basic and utility functions in the proper order to compute apparent, topocentric, or astrometric places of stars or solar system bodies for specific dates and times. If desired, the user can interact exclusively with the supervisory-level functions and not become concerned with the details of the geometry or physical models involved in the computation.

The NOVAS-C source code contains sufficient internal documentation to make the usage clear. Expanded explanations of some of the most frequently called functions are given later in this document. In the Fortran version of NOVAS, some of the basic- and utility-level subroutines are provided in several versions to accommodate users with a need for alternative algorithms. The C version differs from the Fortran version in this regard: only the "standard" version of each algorithm is provided.

The next section of this document (Section 2) provides an overview of the files that constitute NOVAS-C. Section 2 also provides simple instructions for installing and checking the software. Section 3 provides a list and brief description of each NOVAS-C function. Section 4 gives detailed descriptions of some of the most frequently called functions. Finally, Section 5 contains a list of the changes made to update NOVAS-C from version 1.0 to version 2.0.

Throughout this document, **bold text** will be used to refer to file names and *italic text* will be used to refer to function or subroutine names. Variable names or code snippets will be presented in a `typewriter-like font`.

The USNO's Astronomical Application Department maintains a set of NOVAS pages in the Software section of the Department's World Wide Web site. The Astronomical Applications Department's home page is located at *http://aa.usno.navy.mil/AA*.

Important Note

Many changes have been made to NOVAS-C in version 2.0. This includes important changes to the argument lists of many functions, including the supervisory functions. Users familiar with NOVAS-C 1.0 should consult Section 5 for a summary of the changes.

2.0 Getting Started With NOVAS-C

2.1 File Overview

The following files make up the NOVAS-C system:

File name	Description
novas.c	contains all supervisory and utility functions and most basic functions
novas.h	header file for **novas.c** (includes structure definitions and function prototypes)
novascon.c	contains most mathematical and physical constants used by the NOVAS-C system
novascon.h	header file for **novascon.c**
solsys2.c	version of function *solarsystem* that serves as an interface between NOVAS-C and the JPL lunar and planetary ephemerides (see detailed discussion in Section 4)
solsys3.c	version of function *solarsystem* that provides the position and velocity of the Earth or Sun without reference to an external data file (see detailed discussion in Section 4)
solarsystem.h	header file for the "solsys.c" files
readeph0.c	file containing a dummy version of function *readeph*, the highest level call to the USNO minor planet ephemerides software. This file is replaced by **readeph.c** (not supplied with NOVAS-C) when positions of selected minor planet are desired.
jplint.f	Fortran subroutine that serves as the interface between NOVAS-C and JPL's (Fortran) ephemeris access code. For use with the software in **solsys2.c**.

In addition, the following files are provided to assist in validating the installation of NOVAS-C on your local system:

checkout-st.c	main function that calls functions in **novas.c** and **solsys3.c** for the purpose of validating a basic local installation
checkout-st.no	output from the "checkout" application computed at USNO; compare this file with results obtained from your local installation
checkout-mp.c	main function that calls functions in **novas.c**, **solsys3.c**, and the USNO minor planet software for the purpose of validating a local installation of NOVAS-C for use with the minor planet ephemerides
checkout-mp.no	output from the minor planet "checkout" application computed at USNO; compare this file with results obtained from your local installation

2.2 Installation

To install NOVAS-C on your local system, follow the simple instructions given below. These instructions assume that you know how to compile and link C source code on your computer system. Details of the process are dependent on your particular computer system. NOVAS-C has been successfully implemented on PCs running Microsoft Windows, Apple Macintosh systems, and several systems running different flavors of Unix.

1. Copy all NOVAS-C files to a directory on your local system.
2. Compile and link files **checkout-st.c**, **novas.c**, **novascon.c**, **solsys3.c**, and **readeph0.c**. Name the resulting application "checkout".
3. Run the checkout application. Compare the results that you get from "checkout" with the data in file **checkout-st.no**. If the results agree, the installation has probably been successful, but see the important note below.
4. If you plan to use the USNO minor planet ephemerides with NOVAS-C, another checkout program, **checkout-mp.c**, has been supplied. To check the installation of NOVAS-C with the minor planet ephemerides software, repeat step 2, replacing **readeph0.c** provided with NOVAS-C, with **readeph.c**, **allocate.c**, and **chby.c** from the minor planet ephemeris software. Run the resulting application, and compare your results with the contents of **checkout-mp.no**. The ephemeris file for minor planet 2 Pallas (not supplied with NOVAS-C) is required to run this test.

Note that the checkout programs also provide examples of how the NOVAS-C functions are called from an application program.

Important Note

The checkout applications exercise one supervisory function and most, but not all, of the low-level functions in **novas.c**. Also, the checkout applications do not use **solsys2.c**; hence, planetary positions (other than those of the Earth) are not tested. Thus, use of the checkout applications is not a complete test of NOVAS-C. Comparing the results from the NOVAS-C supervisory functions with results from the analogous NOVAS Fortran supervisory functions will constitute a more complete check of your NOVAS-C implementation.

3.0 Function Overview

3.1 Function List

The following functions are contained in file **novas.c**:

Entry name	Level	Purpose
app_star	supervisory	Computes the geocentric apparent place of a star, given its J2000.0 catalog mean place.
topo_star	supervisory	Computes the topocentric apparent place of a star, given its J2000.0 catalog mean place and geographic location of observer.
app_planet	supervisory	Computes the geocentric apparent place of a planet or other solar system body.
topo_planet	supervisory	Computes the topocentric apparent place of a planet or other solar system body, given the geographic location of observer.
virtual_star	supervisory	Computes the "virtual place" of a star, given its J2000.0 catalog mean place.
local_star	supervisory	Computes the "local place" of a star, given its J2000.0 catalog mean place and geographic location of observer.
virtual_planet	supervisory	Computes the "virtual place" of a planet or other solar system body.
local_planet	supervisory	Computes the "local place" of a planet or other solar system body, given the geographic location of observer.
astro_star	supervisory	Computes the astrometric place of a star, given its J2000.0 catalog mean place.
astro_planet	supervisory	Computes the astrometric place of a planet or other solar system body.
mean_star	supervisory	Computes the J2000.0 mean place of a star, given its apparent place.
sidereal_time	supervisory	Computes Greenwich sidereal time, either mean or apparent.
pnsw	supervisory	Transforms arbitrary vector in rotating Earth-fixed (geographic) system to space-fixed (J2000.0) system.
transform_hip	supervisory	Transforms Hipparcos data at epoch J1991.25 to epoch J2000.0 and FK5-style units. To be used only for Hipparcos or Tycho stars with linear space motion.
transform_cat	supervisory	Transforms a star's catalog quantities for a change of epoch and/or equator and equinox.
equ2hor	supervisory	Transforms apparent equatorial coordinates (right ascension and declination) to horizon coordinates (zenith distance and azimuth). Properly accounts for polar motion.
get_earth	utility	Provides barycentric and heliocentric position and velocity of the Earth at a given time.
spin	utility	Rotates vector by angle equal to sidereal time.
wobble	utility	Adjusts Earth-fixed vector for polar motion.
proper_motion	utility	Updates the position vector of a star to allow for its space motion.
bary_to_geo	utility	Changes origin of coordinates from barycenter of solar system to center of mass of Earth.
aberration	utility	Adjusts position vector for aberration of light due to motion of Earth.
precession	utility	Applies precession to position vector.
nutate	utility	Applies nutation to position vector.
sun_field	utility	Adjusts position vector for deflection of light by Sun's gravitational field.
terra	utility	Converts geographic coordinates to geocentric position vector.
vector2radec	utility	Converts position vector to RA and declination.
angle2vector	utility	Converts RA, declination, and distance to a position vector.

starvectors	utility	Converts RA, declination, proper motion, etc., to position and velocity vectors.
nutation_angles	basic	Evaluates nutation series.
fund_args	basic	Computes fundamental arguments (mean elements) of the Sun and Moon.
earthtilt	basic	Provides information on orientation of Earth's axis: obliquity, nutation parameters, etc.
cel_pole	basic	Allows for the specification of celestial pole offsets for high-precision applications.
set_body	basic	Creates a structure of type `body` defining a solar system object based on input parameters.
ephemeris	basic	Retrieves the position and velocity of a body from a fundamental ephemeris.
make_cat_entry	basic	Creates a structure of type `cat_entry` containing catalog data for a star or "star-like" object.
refract	basic	Computes approximate refraction in zenith distance for optical wavelengths.
tdb2tdt	basic	Converts Terrestrial Time (TT or TDT) to Barycentric Dynamical Time (TDB).
julian_date	basic	Computes the Julian date for a given calendar date (year, month, day, hour).
cal_date	basic	Computes a date on the Gregorian calendar given the Julian date.

3.2 Ephemeris Functions

NOVAS-C must have access to a solar system ephemeris. The solar system ephemeris provides NOVAS-C with the heliocentric and barycentric positions and velocities of desired solar system objects referred to the mean equator and equinox of J2000.0. The solar system ephemeris is required even when only precise star positions are needed – in that case, the "desired solar system object" is the Earth. Thus, an ephemeris of the Earth is the minimum requirement.

NOVAS-C accesses ephemerides of solar system objects through function *ephemeris*. This function, as supplied in NOVAS-C 2.0, supports access to an ephemeris of the major solar system bodies (Sun, Moon, and the nine planets), and provides direct support for access to the USNO minor planet ephemerides (USNO/AE98). In order to access the ephemeris of major solar system bodies, function *ephemeris* calls function *solarsystem*. While *solarsystem* has a defined argument list, its inner workings can take any form depending upon the ephemeris that has been selected for use. Users may write their own versions of *solarsystem*, or use either of the two versions provided with NOVAS-C.

See Section 4.18 for detailed information on the ephemeris functions.

3.2.1 Major Planets, Sun, and Moon

Files **jplint.f** and **solsys2.c** contain the software that serves as the interface between NOVAS-C and the JPL lunar and planetary ephemerides, such as DE200, DE405, or DE406. Subroutine *jplint* contains a single call to JPL's Fortran subroutine *pleph*, which

in turn calls other Fortran subroutines in the JPL ephemeris software package. The user must obtain the Fortran ephemeris package, set up the binary, random-access ephemeris file, and link the applicable JPL Fortran code with NOVAS-C. For details, see the discussion of function *solarsystem* version 2 in Section 4.18.1.1.

File **solsys3.c** contains the software (function *solarsystem*, version 3) that provides the position and velocity of the Earth or Sun without reference to an external data file. This version of *solarsystem* is ideally suited for computing coordinates of stars, with errors not exceeding several milliarcseconds. For details, see the discussion of function *solarsystem* version 3 in Section 4.18.1.2.

3.2.2 Minor Planets

In order to access the USNO minor planet ephemerides (USNO/AE98), function *ephemeris* calls function *readeph*. Function *readeph* is part of the USNO/AE98 minor planet ephemeris package and is not part of, or supplied with, NOVAS-C. A dummy version of *readeph* is provided in file **readeph0.c**. The dummy function enables NOVAS-C to be used without the USNO minor planet ephemeris package (i.e. for computing positions of major solar system bodies and "stars" only). To use USNO/AE98 with NOVAS-C, replace file **readeph0.c** provided with NOVAS-C, with **readeph.c**, **allocate.c**, and **chby.c** from the USNO minor planet ephemerides software, when compiling and linking. Minor planet ephemeris files must be created using the utilities provided in the USNO/AE98 package. An ephemeris of major solar system bodies, accessed by function *solarsystem*, is required as well.

3.2.3 Other Bodies and Ephemerides

Users can easily add access to other ephemerides by modifying function *ephemeris*. The code and comments in this function should make the modification self-explanatory.

3.3 Important Data Structures

There are three important data structures used throughout NOVAS-C. They are formally declared in file **novas.h**.

3.3.1 Structure `body`

Structure `body` designates a celestial object.

```
typedef struct
{
   short int type;
   short int number;
   char name[100];
} body;
```

where:

`type`	= type of body = 0 ... major planet, Sun, or Moon = 1 ... minor planet
`number`	= body number For 'type' = 0: Mercury = 1, ..., Pluto = 9, Sun = 10, Moon = 11 For 'type' = 1: minor planet number
`name`	= name of the body (limited to 99 characters)

3.3.2 Structure `site_info`

Structure `site_info` contains data for the observer's location. The atmospheric parameters are used only by the refraction function (*refract*) called from function *equ2hor*. Parameters can be added to this structure if a more sophisticated refraction model is substituted.

```
typedef struct
{
   double latitude;
   double longitude;
   double height;
   double temperature;
   double pressure;
} site_info;
```

where:

`latitude`	= geodetic latitude in degrees; north positive.
`longitude`	= geodetic longitude in degrees; east positive.
`height`	= height of the observer (meters).
`temperature`	= temperature (Celsius).
`pressure`	= atmospheric pressure (millibars)

3.3.3 Structure `cat_entry`

Structure `cat_entry` contains the astrometric catalog data for a star; equator and equinox and units will depend on the catalog. While this structure can be used as a generic container for catalog data, all high-level NOVAS-C functions require J2000.0 catalog data with FK5-type units (shown in square brackets below).

```
typedef struct
{
   char catalog[4];
   char starname[51];
   long int starnumber;
   double ra;
   double dec;
   double promora;
   double promodec;
   double parallax;
   double radialvelocity;
} cat_entry;
```

where:

`catalog[4]`	=	3-character catalog designator (e.g. FK5, HIP, etc.)
`starname[51]`	=	name of star.
`starnumber`	=	integer identifier assigned to star.
`ra`	=	mean right ascension [hours].
`dec`	=	mean declination [degrees].
`promora`	=	proper motion in RA [seconds of time per century].
`promodec`	=	proper motion in declination [arcseconds per century].
`parallax`	=	parallax [arcseconds].
`radialvelocity`	=	radial velocity [kilometers per second].

4.0 Important Functions in NOVAS-C

4.1 APP_STAR

```
short int app_star (double tjd, body *earth, cat_entry *star,

                    double *ra, double *dec)

   PURPOSE:
      Computes the apparent place of a star at date 'tjd', given its
      mean place, proper motion, parallax, and radial velocity for
      J2000.0.

   INPUT
   ARGUMENTS:
      tjd (double)
         TT (or TDT) Julian date for apparent place.
      *earth (struct body)
         Pointer to structure containing the body designation for the
         Earth (defined in novas.h).
      *star (struct cat_entry)
         Pointer to catalog entry structure containing J2000.0 catalog
         data with FK5-style units (defined in novas.h).

   OUTPUT
   ARGUMENTS:
      *ra (double)
         Apparent right ascension in hours, referred to true equator
         and equinox of date 'tjd'.
      *dec (double)
         Apparent declination in degrees, referred to true equator
         and equinox of date 'tjd'.

   RETURNED
   VALUE:
      (short int)
          0...Everything OK.
         >0...Error code from function 'solarsystem'.
```

Discussion:

This function computes the apparent place of a star. The word "star" as used here refers to any object outside the solar system. If the values of `promora`, `promodec`, `parlax`, or `radvel` within structure `star` are unknown (or zero within the errors of measurement), the calling program should set them to zero. For extragalactic objects, these input values should be set to zero. The user's choice of the version of function *solarsystem* determines the value of the argument `earth` that the calling program must supply to *app_star*.

"Loose" catalog data can be assembled into a structure of type cat_`entry` by using function *make_cat_entry*.

4.2 TOPO_STAR

```
short int topo_star (double tjd, body *earth, double deltat,
                     cat_entry *star, site_info *location,

                     double *ra, double *dec)

   PURPOSE:
      Computes the topocentric place of a star at date 'tjd', given its
      mean place, proper motion, parallax, and radial velocity for
      J2000.0 and the location of the observer.

   INPUT
   ARGUMENTS:
      tjd (double)
         TT (or TDT) Julian date for topocentric place.
      *earth (struct body)
         Pointer to structure containing the body designation for the
         Earth (defined in novas.h).
      deltat (double)
         Difference TT (or TDT)-UT1 at 'tjd', in seconds.
      *star (struct cat_entry)
         Pointer to catalog entry structure containing J2000.0 catalog
         data with FK5-style units (defined in novas.h).
      *location (struct site_info)
         Pointer to structure containing observer's location (defined
         in novas.h).

   OUTPUT
   ARGUMENTS:
      *ra (double)
         Topocentric right ascension in hours, referred to true equator
         and equinox of date 'tjd'.
      *dec (double)
         Topocentric declination in degrees, referred to true equator
         and equinox of date 'tjd'.

   RETURNED
   VALUE:
      (short int)
          0...Everything OK.
         >0...Error code from function 'solarsystem'.
```

Discussion:

This function computes the topocentric place of a star (neglecting atmospheric refraction) for the location specified by the argument location, for the time specified by the argument `tjd`. Note that `tjd` is the TT time at which the topocentric place is to be computed. The word "star" as used here refers to any object outside the solar system. If the values of `promora, promodec, parlax, or radvel` within structure `star` are unknown (or zero within the errors of measurement), the calling program should set them to zero. For extragalactic objects, these input values should be set to zero. The difference TT–UT1 (often called ΔT) is passed to the function via argument `deltat`. Values of ΔT are published in the annual *Astronomical Almanac* [11] or can be obtained from the National Earth Orientation Service (NEOS) home page on the World

Wide Web [12]. The user's choice of the version of function *solarsystem* determines the value of the argument `earth` that the calling program must supply to *topo_star*.

"Loose" catalog data can be assembled into a structure of type `cat_entry` by using function *make_cat_entry*.

4.3 APP_PLANET

```
short int app_planet (double tjd, body *ss_object, body *earth,

                      double *ra, double *dec, double *dis)

   PURPOSE:
      Compute the apparent place of a planet or other solar system body.

   INPUT
   ARGUMENTS:
      tjd (double)
         TT (or TDT) Julian date for apparent place.
      *ss_object (struct body)
         Pointer to structure containing the body designation for the
         solar system body (defined in novas.h).
      *earth (struct body)
         Pointer to structure containing the body designation for the
         Earth (defined in novas.h).

   OUTPUT
   ARGUMENTS:
      *ra (double)
         Apparent right ascension in hours, referred to true equator
         and equinox of date 'tjd'.
      *dec (double)
         Apparent declination in degrees, referred to true equator
         and equinox of date 'tjd'.
      *dis (double)
         True distance from Earth to planet at 'tjd' in AU.

   RETURNED
   VALUE:
      (short int)
          0...Everything OK.
         >0...Error code from function 'ephemeris'.
```

Discussion:

This function computes the apparent place of a planet or other solar system body by calling function *ephemeris* to obtain its rectangular coordinates, along with those of the Earth. Other utility- and basic- level functions are also called. The user's choice of ephemerides determines the values to be used in structures `ss_object` and `earth`, which identify the solar system object and the Earth, respectively.

4.4 TOPO_PLANET

```
short int topo_planet (double tjd, body *ss_object, body *earth,
                       double deltat, site_info *location,

                       double *ra, double *dec, double *dis)

   PURPOSE:
      Computes the topocentric place of a planet, given the location of
      the observer.

   INPUT
   ARGUMENTS:
      tjd (double)
         TT (or TDT) Julian date for topocentric place.
      *ss_object (struct body)
         Pointer to structure containing the body designation for the
         solar system body (defined in novas.h).
      *earth (struct body)
         Pointer to structure containing the body designation for the
         Earth (defined in novas.h).
      deltat (double)
         Difference TT(or TDT)-UT1 at 'tjd', in seconds.
      *location (struct site_info)
         Pointer to structure containing observer's location (defined
         in novas.h).

   OUTPUT
   ARGUMENTS:
      *ra (double)
         Topocentric right ascension in hours, referred to true
         equator and equinox of date 'tjd'.
      *dec (double)
         Topocentric declination in degrees, referred to true equator
         and equinox of date 'tjd'.
      *dis (double)
         True distance from observer to planet at 'tjd' in AU.

   RETURNED
   VALUE:
      (short int)
          0…Everything OK.
          >0…Error code from function 'ephemeris'.
```

Discussion:

This function computes the topocentric place of a planet or other solar system body (neglecting atmospheric refraction) for the location specified by the argument location, for the time specified by the argument `tjd`. Note that `tjd` is the TT time at which the topocentric place is to be computed. The difference TT–UT1 (often called ΔT) is passed to the function via argument `deltat`. Values of ΔT are published in the annual *Astronomical Almanac* or can be obtained from the National Earth Orientation Service (NEOS) home page on the World Wide Web. The user's choice of ephemerides determines the values to be used in structures `ss_object` and `earth`, which identify the solar system object and the Earth, respectively.

4.5 VIRTUAL_STAR

```
short int virtual_star (double tjd, body *earth, cat_entry *star,
                        double *ra, double *dec)

   PURPOSE:
      Computes the virtual place of a star at date 'tjd', given its
      mean place, proper motion, parallax, and radial velocity for
      J2000.0.

   INPUT
   ARGUMENTS:
      tjd (double)
         TT (or TDT) Julian date for virtual place.
      *earth (struct body)
         Pointer to structure containing the body designation for the
         Earth (defined in novas.h).
      *star (struct cat_entry)
         Pointer to catalog entry structure containing J2000.0 catalog
         data with FK5-style units (defined in novas.h).

   OUTPUT
   ARGUMENTS:
      *ra (double)
         Virtual right ascension in hours, referred to mean equator
         and equinox of J2000.
      *dec (double)
         Virtual declination in degrees, referred to mean equator
         and equinox of J2000.

   RETURNED
   VALUE:
      (short int)
          0…Everything OK.
         >0…Error code from function 'solarsystem'.
```

Discussion:

See the discussion for function *app_star*. Function *virtual_star* is identical to *app_star* in input arguments and use. Here, however, the output arguments provide the virtual place of the star. The virtual place is essentially the apparent place expressed in the coordinate system of standard epoch J2000.0.

4.6 LOCAL_STAR

```
short int local_star (double tjd, body *earth, double deltat,
                      cat_entry *star, site_info *location,

                      double *ra, double *dec)

   PURPOSE:
      Computes the local place of a star, given its mean place, proper
      motion, parallax, and radial velocity for J2000.0, and the
      location of the observer.

   INPUT
   ARGUMENTS:
      tjd (double)
         TT (or TDT) Julian date for local place.
      *earth (struct body)
         Pointer to structure containing the body designation for the
         Earth (defined in novas.h).
      deltat (double)
         Difference TT(or TDT)-UT1 at 'tjd', in seconds.
      *star (struct cat_entry)
         Pointer to catalog entry structure containing J2000.0 catalog
         data with FK5-style units (defined in novas.h).
      *location (struct site_info)
         Pointer to structure containing observer's location (defined
         in novas.h).

   OUTPUT
   ARGUMENTS:
      *ra (double)
         Local right ascension in hours, referred to mean equator and
         equinox of J2000.
      *dec (double)
         Local declination in degrees, referred to mean equator and
         equinox of J2000.

   RETURNED
   VALUE:
      (short int)
          0...Everything OK.
         >0...Error code from function 'solarsystem'.
```

Discussion:

See the discussion for function *topo_star*. Function *local_star* is identical to *topo_star* in input arguments and use. The local place is essentially the topocentric place expressed in the coordinate system of standard epoch J2000.0.

4.7 VIRTUAL_PLANET

```
short int virtual_planet (double tjd, body *ss_object, body *earth,
                          double *ra, double *dec, double *dis)

   PURPOSE:
      Computes the virtual place of a planet or other solar system body.

   INPUT
   ARGUMENTS:
      tjd (double)
         TT (or TDT) Julian date for virtual place.
      *ss_object (struct body)
         Pointer to structure containing the body designation for the
         solar system body (defined in novas.h).
      *earth (struct body)
         Pointer to structure containing the body designation for the
         Earth (defined in novas.h).

   OUTPUT
   ARGUMENTS:
      *ra (double)
         Virtual right ascension in hours, referred to mean equator
         and equinox of J2000.
      *dec (double)
         Virtual declination in degrees, referred to mean equator
         and equinox of J2000.
      *dis (double)
         True distance from Earth to planet in AU.

   RETURNED
   VALUE:
      (short int)
          0…Everything OK.
         >0…Error code from function 'ephemeris'.
```

Discussion:

See the discussion for function *app_planet*. Function *virtual_planet* is identical to *app_planet* in input arguments and use. Here, however, the output arguments provide the virtual place of the planet. The virtual place is essentially the apparent place expressed in the coordinate system of standard epoch J2000.0.

4.8 LOCAL_PLANET

```
short int local_planet (double tjd, body *ss_object, body *earth,
                        double deltat, site_info *location,

                        double *ra, double *dec, double *dis)

   PURPOSE:
      Computes the local place of a planet or other solar system body,
      given the location of the observer.

   INPUT
   ARGUMENTS:
      tjd (double)
         TT (or TDT) Julian date for local place.
      *ss_object (struct body)
         Pointer to structure containing the body designation for the
         solar system body (defined in novas.h).
      *earth (struct body)
         Pointer to structure containing the body designation for the
         Earth (defined in novas.h).
      deltat (double)
         Difference TT(or TDT)-UT1 at 'tjd', in seconds.
      *location (struct site_info)
         Pointer to structure containing observer's location (defined
         in novas.h).

   OUTPUT
   ARGUMENTS:
      *ra (double)
         Local right ascension in hours, referred to mean equator and
         equinox of J2000.
      *dec (double)
         Local declination in degrees, referred to mean equator and
         equinox of J2000.
      *dis (double)
         True distance from Earth to planet in AU.

   RETURNED
   VALUE:
      (short int)
          0...Everything OK.
         >0...Error code from function 'solarsystem'.
```

Discussion:

See the discussion for function *topo_planet*. Function *local_planet* is identical to *topo_planet* in input arguments and use. The local place is essentially the topocentric place expressed in the coordinate system of standard epoch J2000.0.

4.9 ASTRO_STAR

```
short int astro_star (double tjd, body *earth, cat_entry *star,

                      double *ra, double *dec)

   PURPOSE:
      Computes the astrometric place of a star, given its mean place,
      proper motion, parallax, and radial velocity for J2000.0.

   INPUT
   ARGUMENTS:
      tjd (double)
         TT (or TDT) Julian date for astrometric place.
      *earth (struct body)
         Pointer to structure containing the body designation for the
         Earth (defined in novas.h).
      *star (struct cat_entry)
         Pointer to catalog entry structure containing J2000.0 catalog
         data with FK5-style units (defined in novas.h).

   OUTPUT
   ARGUMENTS:
      *ra (double)
         Astrometric right ascension in hours, referred to mean equator
         and equinox of J2000.
      *dec (double)
         Astrometric declination in degrees, referred to mean equator
         and equinox of J2000.

   RETURNED
   VALUE:
      (short int)
          0...Everything OK.
         >0...Error code from function 'solarsystem'.
```

Discussion:

See the discussion for function *app_star*. Function *astro_star* is identical to *app_star* in input arguments and use. Here, however, the output arguments provide the astrometric place of the star.

4.10 ASTRO_PLANET

```
short int astro_planet (double tjd, body *ss_object, body *earth,

                        double *ra, double *dec, double *dis)

   PURPOSE:
      Computes the astrometric place of a planet or other solar system
      body.

   INPUT
   ARGUMENTS:
      tjd (double)
         TT (or TDT) Julian date for calculation.
      *ss_object (struct body)
         Pointer to structure containing the body designation for the
         solar system body (defined in novas.h).
      *earth (struct body)
         Pointer to structure containing the body designation for the
         Earth (defined in novas.h).

   OUTPUT
   ARGUMENTS:
      *ra (double)
         Astrometric right ascension in hours, referred to mean equator
         and equinox of J2000.
      *dec (double)
         Astrometric declination in degrees, referred to mean equator
         and equinox of J2000.
      *dis (double)
         True distance from Earth to planet in AU.

   RETURNED
   VALUE:
      (short int)
          0...Everything OK.
         >0...Error code from function 'solarsystem'.
```

Discussion:

See the discussion for function *app_planet*. Function *astro_planet* is identical to *app_planet* in input arguments and use. Here, however, the output arguments provide the astrometric place of the planet.

4.11 EQU2HOR

```
void equ2hor (double tjd, double deltat, double x, double y,
              site_info *location, double ra, double dec,
              short int ref_option,

              double *zd, double *az, double *rar, double *decr)

   PURPOSE:
      This function transforms apparent equatorial coordinates (right
      ascension and declination) to horizon coordinates (zenith
      distance and azimuth).  It uses a method that properly accounts
      for polar motion, which is significant at the sub-arcsecond
      level.  This function can also adjust coordinates for atmospheric
      refraction.

   INPUT
   ARGUMENTS:
      tjd (double)
         TT (or TDT) Julian date.
      deltat (double)
         Difference TT (or TDT)-UT1 at 'tjd', in seconds.
      x (double)
         Conventionally-defined x coordinate of celestial ephemeris
         pole with respect to IERS reference pole, in arcseconds.
      y (double)
         Conventionally-defined y coordinate of celestial ephemeris
         pole with respect to IERS reference pole, in arcseconds.
      *location (struct site_info)
         Pointer to structure containing observer's location (defined
         in novas.h).
      ra (double)
         Topocentric right ascension of object of interest, in hours,
         referred to true equator and equinox of date.
      dec (double)
         Topocentric declination of object of interest, in degrees,
         referred to true equator and equinox of date.
      ref_option (short int)
         = 0 ... no refraction
         = 1 ... include refraction, using 'standard' atmospheric
                 conditions.
         = 2 ... include refraction, using atmospheric parameters
                 input in the 'location' structure.

   OUTPUT
   ARGUMENTS:
      *zd (double)
         Topocentric zenith distance in degrees, affected by
         refraction if 'ref_option' is non-zero.
      *az (double)
         Topocentric azimuth (measured east from north) in degrees.
      *rar (double)
         Topocentric right ascension of object of interest, in hours,
         referred to true equator and equinox of date, affected by
         refraction if 'ref_option' is non-zero.
      *decr (double)
         Topocentric declination of object of interest, in degrees,
         referred to true equator and equinox of date, affected by
         refraction if 'ref_option' is non-zero.

   RETURNED
   VALUE:
      None.
```

Discussion:

This function takes the topocentric celestial coordinates of an object and computes the equivalent local horizon coordinates. The function uses a method that properly accounts for polar motion, which is significant at the sub-arcsecond level. Atmospheric refraction can be included in the transformation, and if so, refraction is applied to both sets of coordinates (this can be useful for telescope pointing). Refraction, when requested, is computed by function *refract*.

`ra` and `dec`, the input topocentric right ascension and declination, can be obtained from *topo_star* or *topo_planet.* `tjd` is the TT time at which the topocentric place was computed. The difference TT–UT1 (often called ΔT) is passed to the function via argument `deltat`. Values of ΔT are published in the annual *Astronomical Almanac* or can be obtained from the National Earth Orientation Service (NEOS) home page on the World Wide Web. The coordinates of the pole, `x` and `y`, can be obtained from IERS Bulletins A and B, although `x` and `y` can be set to zero (0.0) if sub-arcsecond accuracy is not needed. (If refraction is requested, sub-arcsecond accuracy is unlikely.)

The height of the observer and meteorological conditions at the observer, contained in structure `location`, are used only for refraction, if `ref_option` is not equal to zero. In this function, the directions `zd` = 0.0 (the zenith) and `az` = 0.0 (north) are considered fixed in the terrestrial frame. Specifically, the zenith is along the geodetic normal, and north is toward the IERS reference pole.

If `ref_option` = 0 (no refraction), then `rar` = `ra` and `decr` = `dec`.

4.12 TRANSFORM_HIP

```
void transform_hip (cat_entry *hipparcos,

                    cat_entry *fk5)

   PURPOSE:
      To convert Hipparcos data at epoch J1991.25 to epoch J2000.0 and
      FK5-style units.  To be used only for Hipparcos or Tycho stars
      with linear space motion.

   INPUT
   ARGUMENTS:
      *hipparcos (struct cat_entry)
         An entry from the Hipparcos catalog, at epoch J1991.25, with
         all members having Hipparcos catalog units.  See Note 1
         below (struct defined in novas.h).

   OUTPUT
   ARGUMENTS:
      *fk5 (struct cat_entry)
         The transformed input entry, at epoch J2000.0, with all
         members having FK5 catalog units.  See Note 2 below (struct
         defined in novas.h).


   RETURNED
   VALUE:
      None.
```

Discussion:

This function takes Hipparcos catalog data, which is given for epoch J1991.25, and transforms it to epoch J2000.0 for use in NOVAS-C functions such as *app_star*. The appropriate units conversion is also performed. Function *transform_cat* is called to perform the epoch transformation.

This subroutine should be used only for Hipparcos or Tycho stars with linear space motion.

Note that radial velocity is not given in the Hipparcos catalog. If a value is not known, set it to zero in the input `hipparcos` structure.

4.13 TRANSFORM_CAT

```
void transform_cat (short int option, double date_incat,
                    cat_entry *incat, double date_newcat,
                    char newcat_id[4],

                    cat_entry *newcat)

   PURPOSE:
      To transform a star's catalog quantities for a change of epoch
      and/or equator and equinox.

   INPUT
   ARGUMENTS:
      option (short int)
         Transformation option
            = 1 … change epoch; same equator and equinox
            = 2 … change equator and equinox; same epoch
            = 3 … change equator and equinox and epoch
      date_incat (double)
         TT Julian date, or year, of input catalog data.
      *incat (struct cat_entry)
         An entry from the input catalog (struct defined in novas.h).
      date_newcat (double)
         TT Julian date, or year, of transformed catalog data.
      Newcat_id[4] (char)
         Three-character abbreviated name of the transformed catalog.

   OUTPUT
   ARGUMENTS:
      newcat (struct cat_entry)
         The transformed catalog entry (struct defined in novas.h).


   RETURNED
   VALUE:
      None.
```

Discussion:

Function `transform_cat` performs mean place to mean place transformations on star catalog data. Only reference data, not observables, are involved. Two dates must be specified: the input data is associated with the first date, and the output data is associated with the second date. Two basic transformations are available:

1. The star's data is updated to account for the star's space motion between the first and second dates, within a fixed reference frame. That is, the *epoch* of the data is changed, but not the equator and equinox.

2. The reference frame within which the star's coordinates and proper motion are expressed is rotated corresponding to precession between the first and second dates. The star's position in space is not changed. That is, the *equator and equinox* of the data are changed, but not the epoch.

These two transformations correspond to `option` = 1 and `option` = 2, respectively. `Option` = 3 requests both transformations, and is the most common case.

This function should be used only for stars with linear space motion; do not use for components of orbit binaries. Also, this function cannot be properly used to bring data from old (pre-FK5) star catalogs into the modern system, because old catalogs were compiled using a set of constants that are incompatible with the IAU (1976) system.

4.14 SIDEREAL_TIME

```
void sidereal_time (double jd_high, double jd_low, double ee,

                    double *gst)

   PURPOSE:
      Computes the Greenwich apparent sidereal time, at Julian date
      'jd_high' + 'jd_low'.

   INPUT
   ARGUMENTS:
      jd_high (double)
         Julian date, integral part.
      Jd_low (double)
         Julian date, fractional part.
      Ee (double)
         Equation of the equinoxes (seconds of time). [Note: this
         quantity is computed by function 'earthtilt'.]

   OUTPUT
   ARGUMENTS:
      *gst (double)
         Greenwich apparent sidereal time, in hours.

   RETURNED
   VALUE:
      None.
```

Discussion:

This function computes Greenwich sidereal time. To obtain the Greenwich mean sidereal time, set input argument `ee` = 0.0. To obtain Greenwich apparent sidereal time, supply the correct value for the equation of the equinoxes (`ee`) which can be computed by calling function *earthtilt*.

The input Julian date may be split into two parts to ensure maximum precision in the computation. For maximum precision, `jd_high` should be set to be equal to the integral part of the Julian date, and `jd_low` should be set to be equal to the fractional part. For most applications the position of the split is not critical as long as the sum `jd_high` + `jd_low` is correct: for example, when used with computers providing 16 decimal digits of precision in double variables, this function will yield values of `gst` precise to better than 1 millisecond even if `jd_high` contains the entire Julian date and `jd_low` is set to 0.0. For ICRS/IERS compatibility when computing apparent sidereal time at millisecond precision or better, you should also use function *cel_pole* and supply the published celestial pole offsets.

For most uses, the input Julian date should be in the UT1 time scale. If the input Julian date is in the TDB time scale, the output must be considered to be 'dynamical' sidereal time.

4.15 PRECESSION

```
void precession (double tjd1, double *pos, double tjd2,

                 double *pos2)

   PURPOSE:
      Precesses equatorial rectangular coordinates from one epoch to
      another.  The coordinates are referred to the mean equator and
      equinox of the two respective epochs.

   INPUT
   ARGUMENTS:
      tjd1 (double)
         TDB Julian date of first epoch.
      pos[3] (double)
         Position vector, geocentric equatorial rectangular coordinates,
         referred to mean equator and equinox of first epoch.
      tjd2 (double)
         TDB Julian date of second epoch.

   OUTPUT
   ARGUMENTS:
      pos2[3] (double)
         Position vector, geocentric equatorial rectangular coordinates,
         referred to mean equator and equinox of second epoch.

   RETURNED
   VALUE:
      None.
```

Discussion:

This function precesses a position vector `pos1` from the equatorial rectangular system of epoch `tjd1` to the equatorial rectangular system of epoch `tjd2`; the resulting vector is `pos2`. The two epochs are completely arbitrary and the transformation is reversible. In typical usage, one of the two epochs will be standard epoch J2000.0, that is, either `tjd1` or `tjd2` will be 2451545.0.

4.16 EARTHTILT

```
void earthtilt (double tjd,

                double *mobl, double *tobl, double *eq, double *dpsi,
                double *deps)

   PURPOSE:
      Computes quantities related to the orientation of the Earth's
      rotation axis at Julian date 'tjd'.

   INPUT
   ARGUMENTS:
      tjd (double)
         TDB Julian date of the desired time

   OUTPUT
   ARGUMENTS:
      *mobl (double)
         Mean obliquity of the ecliptic in degrees at 'tjd'.
      *tobl (double)
         True obliquity of the ecliptic in degrees at 'tjd'.
      *eq (double)
         Equation of the equinoxes in seconds of time at 'tjd'.
      *dpsi (double)
         Nutation in longitude in arcseconds at 'tjd'.
      *deps (double)
         Nutation in obliquity in arcseconds at 'tjd'.

   RETURNED
   VALUE:
      None.
```

Discussion:

This function computes various quantities related to the orientation of the Earth's rotation axis in inertial space at a specific time. The computation involves a call to function *nutation_angles* to evaluate the nutation series. The output values of the last four arguments will correctly reflect the celestial pole offsets if function *cel_pole* has previously been called. A call to *cel_pole* is required for ICRS compatibility.

4.17 CEL_POLE

```
void cel_pole (double del_dpsi, double del_deps)

   PURPOSE:
      This function allows for the specification of celestial pole
      offsets for high-precision applications.  These are added
      to the nutation parameters delta psi and delta epsilon.

INPUT
   ARGUMENTS:
      del_dpsi (double)
         Value of offset in delta psi (dpsi) in arcseconds.
      del_deps (double)
         Value of offset in delta epsilon (deps) in arcseconds.


   OUTPUT
   ARGUMENTS:
      None.

   RETURNED
   VALUE:
      None.
```

Discussion:

This function allows for the specification of celestial pole offsets for high precision (better than 0.1 arcsecond) applications. The offsets are subsequently applied as corrections to the nutation in longitude and nutation in obliquity within *earthtilt*. Thus, *earthtilt*'s output arguments `tobl`, `eq`, `dpsi`, and `deps` will be affected. Since other NOVAS subroutines require *earthtilt* to obtain data related to the Earth's orientation in space, the celestial pole offsets specified here are propagated through the data that the various NOVAS-C functions provide.

Daily values of the celestial pole offsets are published, for example, in IERS Bulletins A and B. The celestial pole offsets effectively correct for errors or incompleteness in the standard precession or nutation models, and are needed for conformity with the ICRS system. If you use *cel_pole*, make sure it is called before any other functions for a given date. Values of the pole offsets that you specify by a call to *cel_pole* will be used by *earthtilt* until you explicitly change them.

4.18 EPHEMERIS

```
short int ephemeris (double tjd, body *cel_obj, short int origin,

                     double *pos, double *vel)

   PURPOSE:
      Retrieves the position and velocity of a body from a fundamental
      ephemeris.

   INPUT
   ARGUMENTS:
      tjd (double)
         TDB Julian date.
      *cel_obj (struct body)
         Pointer to structure containing the designation of the body
         of interest (defined in novas.h).
      origin (int)
         Origin code; solar system barycenter   = 0,
                      center of mass of the Sun = 1.

   OUTPUT
   ARGUMENTS:
      pos[3] (double)
         Position vector of 'body' at tjd; equatorial rectangular
         coordinates in AU referred to the mean equator and equinox
         of J2000.0.
      vel[3] (double)
         Velocity vector of 'body' at tjd; equatorial rectangular
         system referred to the mean equator and equinox of J2000.0,
         in AU/Day.

   RETURNED
   VALUE:
      (short int)
         0    ... Everything OK.
         1    ... Invalid value of 'origin'.
         2    ... Invalid value of 'type' in 'cel_obj'.
         3    ... Unable to allocate memory.
         10+n ... where n is the error code from 'solarsystem'.
         20+n ... where n is the error code from 'readeph'.
```

Discussion:

This function serves as the interface between NOVAS-C and ephemerides of solar system bodies. The version of *ephemeris* that ships with NOVAS-C directly supports an ephemeris of major solar system bodies (such as JPL's DE200, DE405, or DE406), and the USNO minor planet ephemerides (USNO/AE98). The ephemeris of the major bodies is accessed via a call to function *solarsystem* from within function *ephemeris*. The minor planet ephemerides are accessed via a call to *readeph*. (*Note*: both the USNO minor planet ephemerides and the JPL ephemerides are not part of NOVAS-C and must be obtained elsewhere).

It is relatively easy to modify function *ephemeris* to support ephemerides other than the two mentioned above. In function *ephemeris*, there is a `switch` structure controlled

by the value of `type` in a data structure of type `cel_obj`. Currently, two `case`s within the `switch` are defined : `type` = 0 (major bodies via *solarsystem*) and `type` = 1 (minor planets via *readeph*). To support another ephemeris, the user simply defines a new value of `type` and adds another `case` block containing code that accesses the new ephemeris.

Additional information concerning function *solarsystem* is provided in the following sections. A brief description of the USNO minor planet ephemerides and their use in NOVAS-C is given in Section 3.2.2.

4.18.1 SOLARSYSTEM

```
short int solarsystem (double tjd, short int body, short int origin,
                       double *pos, double *vel)

   PURPOSE:
      Provides the position and velocity vectors of a planet or other
      solar system body at a specific time. The origin of coordinates
      may be either the barycenter of the solar system or the center
      of mass of the Sun.

   INPUT
   ARGUMENTS:
      tjd (double)
         TDB Julian date.
      body (short int)
         Body identification number for the solar system object of
         interest;  Mercury = 1,...,Pluto = 9, Sun = 10, Moon = 11.
      origin (short int)
         Origin code; solar system barycenter   = 0,
                      center of mass of the Sun = 1.

   OUTPUT
   ARGUMENTS:
      pos[3] (double)
         Position vector of 'body' at tjd; equatorial rectangular
         coordinates in AU referred to the mean equator and equinox
         of J2000.0.
      vel[3] (double)
         Velocity vector of 'body' at tjd; equatorial rectangular
         system referred to the mean equator and equinox of J2000.0,
         in AU/Day.

   RETURNED
   VALUE:
      (short int)
         0...Everything OK.
         Other values depend upon version in use
```

Discussion:

Function *solarsystem* provides positions and velocities for the major bodies of the solar system. Specifically, this function supplies values for the components of the position vector `pos` and velocity vector `vel` for body `body` at time `tjd`. The vectors computed by `solarsystem` are in the equatorial rectangular coordinate system which is oriented to the mean equator and equinox of standard epoch J2000.0. The vectors are barycentric if `origin` = 0 and heliocentric if `origin` = 1.

There are two different versions of `solarsystem` supplied in NOVAS-C, each with its own internal logic. One uses internally-stored data or series expansions, the other uses

the JPL ephemerides, which exist as external data files. Additional documentation is provided on the following pages for the proper use of each version. The user is free to supply alternative versions, providing that the arguments conform to the above specifications.

The values of the body identification number, `body`, will in general differ from one `solarsystem` version to another; consult the documentation for the specific version in use. Usually, `body` = 1 refers to Mercury, `body` = 2 refers to Venus, `body` = 3 refers to the Earth, etc., but the identification numbers for bodies such as the Sun or Moon vary. Furthermore, some versions of `solarsystem` support only a subset of the major solar system bodies. *The minimum requirement is support for the Earth.* It is also sometimes necessary to distinguish between the Earth and the Earth/Moon barycenter; for computing quantities related to observables (e.g., apparent, topocentric, or astrometric places) it is the position and velocity of the Earth that is required.

4.18.1.1 SOLARSYSTEM, Version 2

```
RETURNED
 VALUE:
    (short int)
       0...Everything OK.
       1...Invalid value of body or origin.
       2...Error detected by JPL software.
```

Discussion:

This version serves as the interface between the Jet Propulsion Laboratory's lunar and planetary ephemeris software and NOVAS-C. The function contains a single call to Fortran subroutine *jplint*, which in turn calls *pleph* and other Fortran subroutines in the JPL ephemeris software package. The user is responsible for obtaining the Fortran ephemeris code and data, setting up the binary, random-access ephemeris file, and linking the JPL Fortran code with NOVAS-C. See the Implementation Notes below.

The body identification numbers to be used with this version are: Sun, `body` = 10; Mercury, `body` = 1; Venus, `body` = 2; Earth, `body` = 3; Mars, `body` = 4; Jupiter, `body` = 5; Saturn, `body` = 6; Uranus, `body` = 7; Neptune, `body` = 8; Pluto, `body` = 9; Moon, `body` = 11.

Implementation Notes:

In order to use NOVAS-C with *solarsystem* version 2, you must first obtain the export planetary ephemeris package from JPL. Be sure to choose an ephemeris whose coordinates are oriented to the mean equator and equinox of standard epoch J2000.0, such as DE200, DE405, or DE406. The export package is available over the Internet from the anonymous ftp server *navigator.jpl.nasa.gov/ephem/export* and consists of several large ASCII data files and software provided in the form of Fortran source code. An installation guide is also included. Alternatively, the export package is also available on CD-ROM from Willmann-Bell, Inc. (*www.willbell.com*) for a modest fee. The installation process consists of converting (large) files of ASCII ephemeris data to binary, direct-access form using a supplied utility program. Then, the binary file is verified using another utility program and a file of comparison data. If the verification process is successful, the ephemeris file is ready to use. The ephemeris data is obtained from the binary file by calling the access subroutines provided in the export package.

Important Note
Over the years, there have been several versions of the JPL export ephemeris software. The following discussion specifically refers to the software version available on the "JPL Planetary and Lunar Ephemerides" CD-ROM ©1997.

Version 2 of *solarsystem* (C code) obtains ephemeris data from the binary file by calling Fortran subroutine *jplint*, which is part of the NOVAS-C package. Subroutine

jplint, in turn, calls JPL subroutine *pleph* (Fortran code). The C function *solarsystem* has a few features that make it possible for it to exchange data with the Fortran subroutine *jplint*. First, all of the C arguments of the call to *jplint* are addresses, since Fortran uses call by address instead of call by value for arguments of subroutines. Second, all of the integer arguments in the call are designated as type `long int` in the C function to match the Fortran `INTEGER` default. The `DOUBLE PRECISION` arguments in the subroutine are designated as type `double` in the C function.

Probably the biggest hurdle in implementing version 2 of *solarsystem* will involve the proper compiling and linking of the mixed-language files. The procedures will be specific to your computing platform; therefore, you will have to consult your compiler manual for detailed instructions. The following instructions are offered only as a guideline – they provide a specific example of how the mixed-language files were successfully handled on an IBM RISC System 6000 Unix workstation.

1. Create a single file with all of the JPL Fortran ephemeris access subroutines. Name it **jplsubs.f**.

2. Compile the Fortran files without invoking the linkage editor. This creates the object file **jplsubs.o** and **jplint.o**. The Fortran compiler/linker is `xlf`.

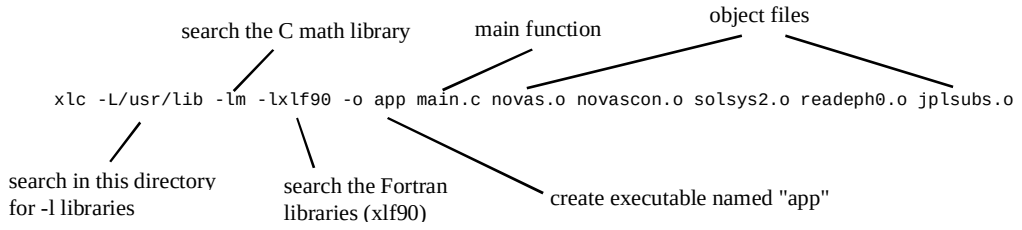
```
xlf -c jplsubs.f jplint.f
```

3. Compile, again without invoking the linkage editor, the C files **novas.c**, **novascon.c**, **solsys2.c,** and **readeph0.c**. This creates the object files **novas.o**, **novascon.o**, **solsys2.o**, and **readeph0.o**:

```
xlc -c -lm novas.c novascon.c solsys2.c readeph0.c
```

The C compiler/linker is `xlc`. The `-lm` option specifically searches the math library.

4. Finally, compile the main function and link it with the object files:



In this example, the resulting executable file is named **app**. Note especially the use of the `-l` option to force the C compiler/linker to search the Fortran libraries for unresolved references.

Important Note

It is strongly recommended that results obtained from your specific implementation of NOVAS-C and *solarsystem* version 2 be checked by comparing to corresponding values published in the *Astronomical Almanac*, or by comparing to results obtained from the Fortran version of NOVAS using subroutine *SOLSYS* version 2.

4.18.1.2 SOLARSYSTEM, Version 3

```
RETURNED
VALUE:
   (short int)
      0...Everything OK.
      1...Input Julian date ('tjd') out of range.
      2...Invalid value of 'body'.
```

Discussion:

This version of *solarsystem* provides the position and velocity of the Earth or Sun without reference to any external data file. The heliocentric position and velocity of the Earth are computed by evaluating trigonometric series. When barycentric positions and velocities are required, a number of somewhat crude approximations are involved; therefore, barycentric positions and velocities computed by this version of *solarsystem* are less accurate than heliocentric positions and velocities. The resulting errors should be less than the following values:

Maximum error in heliocentric positions:	6×10^{-6} AU
Maximum error in heliocentric velocities:	8×10^{-7} AU/day
Maximum error in barycentric positions:	7×10^{-4} AU
Maximum error in barycentric velocities:	2×10^{-6} AU/day

When this version of *solarsystem* is used in the computation of the apparent place of the Sun, it should contribute less than 2 arcseconds error. When this version of *solarsystem* is used in the computation of apparent places of stars, it should contribute less than 2 milliarcseconds error. This error assessment applies to the interval 1800–2050.

Note: This version of *solarsystem* calls several other functions in the NOVAS-C package.

The body identification numbers to be used with this version are: Sun, `body` = 0, `body` = 1, or `body` = 10; Earth, `body` = 2 or `body` = 3.

5.0 Changes in NOVAS-C Version 2.0 From Version 1.0

- change the argument lists of the highest-order functions: body designations are now structures instead of simple (short) integers. This accommodates a wider range of body types.

- added direct support for USNO minor planet ephemerides (USNO/AE98) and indirect support for other ephemerides with new function *ephemeris*.

- added support for latest (1997 CD-ROM) version of the JPL solar system ephemeris software in **solsys2.c** and **jplint.c**

- incorporated IAU 1994 (IERS 1996) definition of the sidereal time (implemented as a change to the calculation of the equation of the equinoxes in function *earthtilt*).

- generalized data structure used to contain star catalog data (`cat_entry` in **novas.h**)

- added new function *make_cat_entry* that creates a `cat_entry` data structure from "loose" star catalog data.

- added two new functions (*transform_hip* and *transform_cat*) to support use of non-FK5 data in NOVAS-C. Specifically, *transform_hip* supports use of Hipparcos data.

- added two new functions (*equ2hor* and *refract*) to support transformation of equatorial coordinates to horizon coordinates, with refraction optional.

- created new function, *fund_args*, which contains the fundamental arguments of the nutation series.

- created global variables `PSI_COR` and `EPS_COR` and function *cel_pole* to provide observed celestial pole offsets.

- changed names of several low-level functions to be more descriptive:
  geocentric to *bary_to_geo*
  nutation to *nutation_angles*
  convert_tdb2tdt to *tdb2tdt*

- update function *tdb2tdt* using expressions for mean elements referred to J2000 epoch and reference system.

- changed type of function precession from `short int` to `void`

- TT time scale is used interchangeably with TDT time scale.

- constants:
  - moved `f` and `omega` from function `terra` to file **novascon.c**.
  - updated value of `C` in AU/day.
  - updated value of `OMEGA`.
  - updated value of `T0`.
  - removed `PI` from **novascon.c** to avoid conflict with definition of `PI` in Linux **math.h**.

- changed name of `sun` function in **solsys3.c** to `sun_eph` to fix problem on Sun Unix systems.

- updated prologs and documentation.

- cosmetic changes.

6.0 Acknowledgements

Thomas K. Buchanan, working as part of the U.S. Naval Observatory/Naval Research Laboratory Optical Interferometer team, did the initial conversion of many of the NOVAS Fortran subroutines to C.

William T. Harris of the Astronomical Applications Department of the U.S. Naval Observatory was largely responsible for completing the conversion of the NOVAS Fortran subroutines to C, and for NOVAS-C Version 1.0.

David Buscher, James Hilton, Christian Hummel, and Sandra Martinka, users of preliminary versions of the NOVAS-C package, provided valuable comments and suggestions.

7.0 Notes, References, And URLs

1. Kaplan, G. (1990), *Bulletin of the American Astronomical Society*, Vol. 22, pp. 930-931.
2. See explanation and references at *http://aa.usno.navy.mil/AA/faq/docs/ICRS_doc.html*
3. ESA, 1997, The Hipparcos and Tycho Catalogues, ESA SP-1200. See also *http://astro.estec.esa.nl/Hipparcos/catalog.html*
4. See *http://aries.usno.navy.mil/ad/act/act.html*
5. See *http://maia.usno.navy.mil/ICRF/*
6. The JPL ephemerides are available free for download from ftp server *navigator.jpl.nasa.gov/ephem/export* or on CD-ROM for a modest fee from Willmann-Bell, Inc. *http://www.willbell.com/software/jpl.htm*
7. See *http://hpiers.obspm.fr/*
8. Hilton, J. (1999), *Astronomical Journal*, Vol. 117, pp. 1077-1086. See also *http://aa.usno.navy.mil/hilton/ephemerides/asteroid_ephemerides.html*
9. Kaplan, G., et al. (1989) *Astronomical Journal*, Vol. 97, p. 1197.
10. Kaplan, G. (1998) *Astronomical Journal*, Vol. 115, p. 361
11. *The Astronomical Almanac* and other related publications are described at *http://aa.usno.navy.mil/AA/publications/docs/almanacs.html*
12. See *http://maia.usno.navy.mil/*